THE COMMON BUSINESS COMMUNICATION LANGUAGE

John McCarthy
Computer Science Department
Stanford University
Stanford, CA 94305
jmc@cs.stanford.edu
http://www-formal.stanford.edu/jmc/

1999 May 7, 1:55 p.m.

Abstract

This paper, written in 1975, published in 1982 [McC82] in the proceedings of a conference whose title was in German, seems to be worth reviving in 1998, because some of its ideas are new in 1998 (I'm told) and are relevant to the new interest in electronic commerce.

1 The Problem of Inter-computer Communication

Here are some ideas about the value of a *common business communication language* (CBCL for short) and what its characteristics might be. Besides its practical significance, CBCL raises issues concerning the semantics of natural language.

The need for such a language was suggested to me by an article by Paul Baran [Bar67].[1] In this article, Baran envisaged a world of the future in which

[1] 1998 footnote: The question became salient for me when I attended a DARPA meeting in 1969 at which the Navy's Fleet Data System was described.

companies would be well equipped with on-line computer systems. The inventory control computer of company A would write on the screen of a clerk in the purchasing department a statement that 1000 gross of such-and-such pencils were needed and that they should be purchased from company B. The clerk would turn to her typewriter and type out a purchase order. At company B another clerk would receive the purchase order and turn to her terminal and tell the computer to arrange to ship the pencils. Eliminating both clerks by having the computers speak directly to each other was not mentioned. Perhaps the author felt that he was already straining the credulity of his audience.

Suppose we wish to eliminate the clerks by having the computers speak directly to each other. *What are the requirements?*

2 Requirements

First, computers do communicate directly now (1975). In the late 1950s the Social Security Administration announced a format for IBM seven channel magnetic tape on which it was prepared to receive reports of earnings and payroll deductions. Note the limitations: (i) magnetic tapes are mailed rather than direct electronic communication—admittedly entirely appropriate in this case. (ii) A single fixed kind of message with a fixed set of parameters exists for each report. (iii) There is only one receiver of information which can dictate the format. Today information is often exchanged electronically among computer systems belonging to different organizations, and this is usually by specific treaty betwen the two organizations, but sometimes a group that will be communicating has agreed on formats. An example is the U.S. Navy's Fleet Data System for exchanging information among ships about what their radars and other sensors can see so that each ship can have the full radar picture acquired by the whole fleet. In connection with the extension of the system to NATO, it was completely redesigned, and on a designated day, all users switched to the new system.

Our goal is more ambitious in the following respects:

1. A common language is to be adopted that can express business communications. For example, requests for price quotations, offers to buy and sell, queries about delivery times and places, inquiries about the status of delayed orders, references to standard commercial legal agreements. If possible, the same language should with only different primitives suffice to communi-

cate the Navy's or the FAA's radar information or a request from one state's department of motor vehicles to another's for a list of a person's traffic convictions.

2. Any organization should be able to communicate with any other without pre-arrangement over ordinary dial-up telephone connections. Of course, this requires authentication procedures and verification of authorization procedures, but let us not be unduly distracted by the security aspects of computing lest we end up with a secure method of communication and nothing to say.[2]

3. The system should be open ended so that as programs improve, programs that can at first only order by stock numbers can later be programmed to inquire about specifications and prices and decide on the best deal. This requires that each message be translatable into a human-comprehensible form and that each computer have a way of referring messages it is not yet programmed to understand to humans. When a new type of message is to displace an old one, the programs should send both until all the receivers can understand the new form. Thus the crises of cutover days, as in the naval example, could be eliminated.[3]

4. CBCL is strictly a communication protocol. It should not presuppose any data-base format for the storage within machines of the information communicated, and it should not presuppose anything about the programs that use the language. Each business using the language would have a program designed to use the particular part of CBCL relevant to its business communications. Thus CBCL presupposes nothing about the programs that decide when to order or what orders to accept.

5. CBCL is not concerned with the low-level aspects of the message formats, i.e., what kinds of bit streams and what kinds of modems, except to remark that the system should avoid traps in these areas, and the users should be able to change their systems asynchronously. Presumably CBCL would use the same low level protocols used for more simple inter-business communications like person-to-person messages and file transfer.

We do not have a final proposal but here are some ideas:

1. The messages are lists of items punctuated by parentheses. The lead item of each list identifies the type of message and is used to determine how to interpret the rest. The items may be either sublists or atoms. If an item

[2] 1998 footnote: Emphasizing security over content is still a problem.

[3] 1998 footnote: I didn't see this in the ICE literature.

is a sublist, its first element tells how to interpret it. Atoms are binary numbers of say 32 bits. A dictionary tells what each means. Other forms of data may be used provided they are demarcated by appropriate punctuation and provided they are pointed at from lists that tell how they are to be interpreted.[4]

2. Here are some examples:

a. (REQUEST-QUOTE (YOUR-STOCK-NUMBER A7305) (UNITS 100))

b. (REQUEST-QUOTE (PENCILS #2) (GROSS 100))

[1998: The above two examples correspond directly to what has been proposed for ICE apart from the names of the structures.]

c. (REQUEST-QUOTE (ADJECTIVE (PENCILS #2) YELLOW) (GROSS 100))

[1998: The point of ADJECTIVE is that a program not understanding YELLOW could nevertheless understand that #2 pencils were called for, and could reply that they don't have any pencils, if that were so.]

d. (WE-QUOTE (OUR-STOCK-NUMBER A7305) (QUANTITY 100) (DELIVERY-DATE 3-10-77) (PRICE $1.00))

[1998: This is also standard today.]

---

[4] 1998 footnote: This list format is isomorphic to XML but simpler. The first example below translated into XML becomes

a.

```
<REQUEST-QUOTE>
        <YOUR-STOCK-NUMBER> A7305 </YOUR-STOCK-NUMBER>
        <UNITS> 100 </UNITS>
</REQUEST-QUOTE>,
```

with no obvious benefit over

```
(REQUEST-QUOTE
    (YOUR-STOCK-NUMBER A7305)
    (UNITS 100)).
```

e. (PLEASE-SAY (IOTA (X) (AND (RED X) (PENCIL X))))

[1998: This uses the Russell description operator, essentially corresponding to the English word "the". [1998: That's not such a good example. Here's a better one using the Hilbert ϵ (epsilon) symbol: (PLEASE-RESERVE (EPSILON (X) (AND (IS-FLIGHT X) (DEPARTS MONDAY) (ARRIVES (BEFORE WEDNESDAY))))). $(\epsilon x)P(x)$ stands for "an x such that $P(x)$. If you don't like ϵ, you can write (AN (X)...,etc. This raises a general expression about variable binders. CBCL proposes to handle them in a standard way, namely (<binder> (<variables> <body>), i.e. all binders including those to be invented in the future are handled the same. The logical operators AND, OR, and NOT are also to be standard where used at all.]

It appears that some items may require a variable number of modifiers.

As a toy example, imagine writing conventions that would permit any Monopoly-like game to be played by independently written programs. Suppose that the moves are communicated to a referee who receives requests to roll the dice and returns information about what squares the pieces landed on and what 'chance' cards were drawn. The programs would communicate offers to buy and sell directly to each other and to the 'banker'.

CBCL should have an important property enunciated by Chomsky in his *Reflections on Language* as a characteristic of human language. (Linguists tell me that whether natural languages have it is controversial; but whether they do or don't, CBCL shall have it.) The principle (reworded considerably) is that no grammatical position should require an identifier or a number per se but should allow a phrase. For example, instead of requiring a stock number, an expression designating the stock number, such as 'the same stock number as last week' or 'the new stock number of the item that was formerly stock number 2531'. We don't really mean these English phrases but rather whatever CBCL expression translates into them. [5]

3 CBCL and natural language

Developing an expressive CBCL has proved unexpectedly difficult. Even concentrating on the idea of a purchase order doesn't easily lead to defining formats that permit expressing all that should be possible to include in a purchase order. The problem is that every aspect of the purchase order

[5] This isn't part of the ICE specification as far as I can see.

such as the delivery method or the terms of payment seems to admit infinite variation and elaboration. It is a semantic feature of natural language that this elaboration is possible. The problems do not at all stem from the rigid list syntax of CBCL, which after all resembles the result of parsing a natural language text. The problems are in the semantics, i.e., in specifying what should be expressible.

This suggests that the problem of formalizing what is expressible in natural language can and should be studied entirely separately from the syntax. In addition it suggests that putting natural language front ends on computer programs often entirely misses the key problems of natural language. Namely, before the natural language front end is attached, the programmer has already decided what things shall be sayable, and they are usually things that can readily be said in a pre-existing input-output system. But if we are right, the most difficult problems in making a computer use language involve deciding what is to be sayable.

For example, consider some possible specifications of the method of delivery.

1. By air excluding Capital Airlines.

2. By air excluding Capital Airlines provided this doesn't delay the shipment more than a day.

3. As soon as possible without incurring extra charges.

4. By truck complying with the rules on shipment of explosives (even though the present shipment isn't classified as explosive).

5. By truck making sure our competitor doesn't learn the size and model number of the item shipped.

Unfortunately, these few examples do not show the scope of the problem.

In order to make sense of expressions like those in the above examples programs that use CBCL will have to be capable of non-monotonic reasoning. Namely, they will need to be able to give the most standard interpretations of the messages compatible with what has been said explicitly. Circumscription has been proposed as a tool for this.

4 1998: Advice for XML, W3 and ICE

This summarizes and extends remarks made in the 1998 footnotes.

1. It is important to keep the language incrementally extendable. Many

extensions will add detail to messages so that less human intervention will be required.

2. Lisp notation is better. Oh, well, the committees have decided otherwise. Anyway XML is isomorphic to the subset of Lisp data where the first item in a list is required to be atomic.

3. Lisp data provides the additional generality that the first item of the list may itself be compound and have to be evaluated to determine how the rest of the list is to be interpreted. I don't know whether this would get much use in CBCL or XML.

4. Use modifiers like ADJECTIVE. Thus (ADJECTIVE FOO YELLOW) means a yellow FOO. However, a program not yet equipped to understand YELLOW but which can understand FOO may be able to do something useful with the information, given the convention that (ADJECTIVE x y) is a kind of y. Many English adjectives are not used this way, and we propose only to use such proper adjectives.

5. All expressions that may be taken apart should have the standard syntax at least as an alternative to special string syntaxes. The Lisp systems don't do this properly, e.g. with time strings, and I notice that the W3 draft doesn't either. In Lisp a time has the string format exemplified by Wed Nov 11 12:58:28 1998. It should also allow the format (TIME <weekday> <month> <hour> <minute> <second>). The operators on such string should have the names they would have in the list or XML format.

6.

References

[Bar67] Paul Baran. The future computer utility. *The Public Interest*, (8):75–87, 1967.

[McC82] John McCarthy. Common Business Communication Language[6]. In Albert Endres and Jürgen Reetz, editors, *Textverarbeitung und Bürosysteme*. R. Oldenbourg Verlag, Munich and Vienna, 1982.

[6]http://www-formal.stanford.edu/jmc/cbcl.html

/@steam.stanford.edu:/u/ftp/jmc/cbcl.tex: begun 1996 May 14, latexed 1999 May 7 at 1:55 p.m.